Plunner

Score Project: MeetMe

Product: Plunner

Score project summary report

Table of contents

Introduction

Abstract

Plunner is a new meeting planner addressed to small and medium business organisations. It uses industry standard technologies like CalDAV.

The key feature of Plunner, with respect to existing ones, is the optimization function.

Plunner allows the optimization of all meetings of a company maximizing the number of employees attending them.

Optimizing human resources and number of participants is an essential part in business companies.

Keywords

Operations Research
Optimization
CalDAV
Meeting planner
Organisations
Companies
Business

Document Purpose

This document is intended to illustrate the features, the concepts and the development process behind the project called “Plunner”. The focus is on the ideas that inspired the project, the requirements, the design and the implementation of the project, the software engineering strategy used. Besides, we describe how it has been tested and how the experience of its development was from the point of view of the team behind it.

Background

In businesses and organizations meetings are essential for producing value and guaranteeing the quality and correctness of a product or a service. Organizing these meetings can be tedious and time consuming since different needs have to be harmonized in order to find a suitable date and time. In addition to that, the tools usually used to organize meetings are not powerful or intuitive enough for the needs of modern businesses or organizations.

Objectives

Plunner is a web application which aims to solve the problems underlined above, by providing a flexible and intuitive way to plan and organize meetings for businesses and organizations. Plunner has optimization and simplicity in mind, so that:

- Meetings can be planned by importing schedules from external calendaring services using CalDAV or by composing in-app schedules
- Meetings times and dates are determined automatically by optimizing (maximizing) the number of participants
- All the functionalities of the application can be accessed using a simple, intuitive and

responsive web interface

Resources

We created a real domain associated with a web-server for our application.

For use the application, install it or contribute to it refer http://plunner.com/. For the repositories associated with the project refer https://github.com/dsd-meetme.

Team

The team is distributed across different countries and universities and it is composed by:

Claudio Cardinale, Politecnico di Milano University, claudio.cardinale@mail.polimi.it
Denis Kuryshov, Politecnico di Milano University, jaibird2493@gmail.com
Jean Barré, Politecnico di Milano University, jeanperrin.barre@mail.polimi.it
Giorgio Pea, Politecnico di Milano University, giorgio.pea@mail.polimi.it
Mihovil Vinković, Mälardalen Högskola University, mihovilvinkovic@gmail.com
Emil Silađi, Mälardalen Högskola University, emil.siladi@gmail.com

Definitions, Acronyms, Terms

Keyword	**Definition**
Registered Organization	An organization that has registered to Plunner
Member	A member of a registered organization. This member has been registered into Plunner by its organization
Planner of a team g	A particular Member that can manage the planning of meetings for team g(one planner per team)
Team	A set of Members of a registered organization. This set can have no counterpart in the real life, it's just a schema Plunner imposes to simplify things
Schedule of a Member	A set of time slots relative to a given period(a month for example) in which the Member is busy
Sunday Midnight	Sunday at 00:00 that is to say one minute after the 23:59 of Saturday

Acronym or abbreviation	**Definition**
GLPK	GNU Linear Programming Kit
LDAP	Lightweight Directory Access Protocol
AWS	Amazon Web Services
JWT	JSON Web Tokens
LTS	Long term support
HTTPS	HTTP over SSL
HTTP	Hypertext Transfer Protocol

Requirements

Actors

The actors involved in the functional requirements of the project are:
Organization, Registered organization, Member, Planner of a team g (see Definitions sections for more information)

Functional Requirements

Here we define functional requirements (Strikethrough requirements are not done). To decide the right priority we have defined a simple, but systematic, mathematical method.

Score

For each requirement is assigned

- An effort in terms of hours
- A ROI with a fixed scale [0-100]
- The score calculated as: ROI^2/effort

For each user story the sum of the score of its requirements (TOTSCORE) is computed and the following priorities are assigned as a consequence

- 1000 <= TOTSCORE → High priority(HIGH)
- 500 <= TOTSCORE < 1000 → Medium priority(MED)
- TOTSCORE < 500 → Low priority(LOW)

If a requirement has a ROI >= 90 then it is considered required.

Requirements

ID	STORY			TOT SCORE
1	**As an organization I want to register and login to the service**			**5950** **HIGH**
REQ ID	REQUIREMENT DESCRIPTION	EFFORT [h]	ROI [0-100]	SCORE
1	The Service must allow an organization to signup via email and password	5	100	2000
2	The Service must allow a registered organization to login via email and password	5	100	2000
3	The Service must allow a registered organization to recover its login credentials	5	80	1280

~~4~~	~~The Service must allow a registered organization to use a two factor authentication~~	~~10~~	~~30~~	~~90~~
5	The Service must allow a remember me functionality	10	20	40
~~6~~	~~The Service must assign to a just registered organization a service specific domain~~	~~15~~	~~90~~	~~540~~
~~7~~	~~The service must verify company email~~	~~5~~	~~35~~	~~245~~
8	The service must allow to see and edit personal information	5	70	980
9	the service must allow to change the password	5	70	980

ID	STORY			TOT SCORE
2	**As a registered organization I want to register/Import and to manage my employees**			**1722.5** **HIGH**
REQ ID	REQUIREMENT DESCRIPTION	EFFORT [h]	ROI [0-100]	SCORE
1	The Service must allow a registered organization to register to the service, within the organization's context, its employees with their corporate email address and a password	25	100	400
~~2~~	~~The Service must allow a registered organization to import to the service, within the organization's, its employees using a standard protocol(LDAP)~~	~~35~~	~~70~~	~~140~~
3	The Service must allow a registered organization to add and remove to the service, within the organization's context, employees	40	90	202.5
4	The Service must allow a registered organization to automatically send via mail its service specific domain to every new registered employee. This email will contain also the password the organization chose for the employee	5	70	980

ID	STORY			TOT SCORE

3	**As a registered organization I want to create and manage my teams**			**2516.666** **HIGH**
REQ ID	REQUIREMENT DESCRIPTION	EFFORT [h]	ROI [0-100]	SCORE
1	The Service must allow a registered organization to organize its employees in teams	20	90	405
2	The Service must allow a registered organization to choose for each teams one of its member to become a planner	10	85	722.5
3	The Service must allow a registered organization to change the role of the members of each team	15	80	426.6666
4	The Service must allow a registered organization to change the name and the composition of each team	20	80	320
5	The Service must allow a registered organization to delete teams	10	75	562.5
~~6~~	~~The Service must allow a registered organization to merge teams or split teams in subteams~~	~~20~~	~~40~~	~~80~~

ID	STORY			TOT SCORE
4	**As a Member I want to login to the service within the context of the organization**			**1223.3333** **HIGH**
REQ ID	REQUIREMENT DESCRIPTION	EFFORT [h]	ROI [0-100]	SCORE
1	The Service must allow Members/planners to login within the context of their organization using the corporate mail and the password received when their were registered	30	100	333.33333
2	The Service must allow Members/planners to recover their login credentials	10	70	490
3	The Service must allow the remember me functionality for Members/planners	10	40	160
4	~~The Service must allow a Member/planner to use a two factor authentication(this decision is made by the~~	~~15~~	~~60~~	~~240~~

	~~Member/planner's organization)~~			
5	The service must allow to see and edit personal information	5	60	720
6	THe service must allow to change the password	5	60	720

ID	STORY			TOT SCORE
5	**As a Member i want to upload from a file/import from an external service my schedule and i want to manage it**			**1582.8333 HIGH**
REQ ID	REQUIREMENT DESCRIPTION	EFFORT [h]	ROI [0-100]	SCORE
1	The Service must allow Members to upload schedules as files of a standard format	20	70	245
2	The Service must allow Members to import schedules from external services using, when is possible, as standard protocol like CalDav(otherwise using specific apis)	30	90	270
3	The Service must allow Members to keep schedules imported from external services in sync with those services	30	80	213.33333
4	The Service must allow Members to keep track of the schedules uploaded/imported	50	60	72
5	The Service must allow Members to remove schedules	10	60	360
6	The Service must allow Members to enable or disable schedules	10	65	422.5

ID	STORY			TOT SCORE
6	**As a Member I want to indicate the time slots when i'm free**			**675 MEDIUM**
REQ ID	REQUIREMENT DESCRIPTION	EFFORT[H]	ROI [0-100]	SCORE
1	The Service must allow Members to indicate in their profile (in a calendar like view, using dragging) time slots in which they're busy.	30	90	270
2	The Service must allow to change the busy time slots	20	90	405

ID	STORY			TOT SCORE
7	**As a Member I want to export a planned meeting of my teams to my calendars**			**960 MEDIUM**
REQ ID	REQUIREMENT DESCRIPTION	EFFORT [H]	ROI [0-100]	SCORE
~~1~~	~~The Service must be able to write a planned meeting to external calendaring services~~	~~15~~	~~60~~	~~240~~
~~2~~	~~The Service must be able to generate a file with the Member general schedule. This schedule will contain also the planned meeting~~	~~5~~	~~60~~	~~720~~

ID	STORY			TOT SCORE
8	**As a Member I want to receive notifications about meetings planned for my teams**			**386.66666 LOW**
REQ ID	REQUIREMENT DESCRIPTION	EFFORT [H]	ROI [0-100]	SCORE
1	The Service must be able to send to Members notifications about the status of a meeting to be planned in which they're involved	15	70	326.66666
2	~~The Service must be able to send to Members notifications about the necessity of an input to help planning a meeting~~	~~15~~	~~30~~	~~60~~

ID	STORY			TOT SCORE
9	**As a planner I want to plan meetings for a team**			**1719.6666 HIGH**
REQ ID	REQUIREMENT DESCRIPTION	EFFORT [h]	ROI [0-100]	SCORE
1	The Service must allow planners to plan a meeting, specifying a title, brief description, temporal interval and some tags	30	100	333.33333
~~2~~	~~The Service must allow planners to plan a meeting specifying which Members of their teams are required and/or invited~~	~~20~~	~~80~~	~~320~~

~~3~~	~~The Service must support meetings to be planned with a given frequency(for example a team has to plan a meeting every two weeks)~~	~~20~~	~~70~~	~~245~~
4	The Service must support meetings to be planned taking in consideration of the different time zones of the Members	10	80	640
5	The Service must support meetings to be planned trying to maximize the number of Members that have to partecipate to different meetings in the same temporal interval (e.g i belong to 2 teams and their planners plans two meetings in the same week)	50	80	128
~~6~~	~~The Service must support meetings to be planned with given constraints (for example, the number of invited Members has to be > k)~~	~~30~~	~~40~~	~~53.333333~~

ID	STORY			TOT SCORE
10	**As a planner I want to manage meetings**			**1645 HIGH**
REQ ID	REQUIREMENT DESCRIPTION	EFFORT [h]	ROI [0-100]	SCORE
1	The Service must allow planners to drop a meeting to be planned or already planned	10	80	640
2	The Service must allow planners to change the title, the description, the Members of, the place of a meeting to be planned or already planned	20	80	320
3	The Service must allow planners to change the temporal interval of a meeting to be planned	10	80	640
~~4~~	~~The Service must allow planners to change the constraints of a meeting to be planned~~	~~20~~	~~30~~	~~45~~

Non functional Requirements

In theory the team would have wanted to implement an high performance system with data redundancy, high uptime, secure protocols and authentication, however for economical and time reasons the team did not manage to achieve these requirements except for the use of a secure authentication system based on JWT

The development process

SCRUM overview

The team has adopted the SCRUM software development methodology. SCRUM addresses the problem of requirements volatility and unpredicted changes in a software project through a series of tools, ideas and organization rules.
This methodology has been chosen for the following reasons:

- It organizes the development process in a way that the problem of requirements volatility and unpredicted changes to the software project is addressed in a robust and relatively simple way
- It promotes team working and communication and so it's great to know new people from different cultures and countries
- It imposes a systematic way to approach to the different steps of the development process through sprints, meetings and sprint backlogs
- It is document oriented which is useful for the future careers of the members of the project team and for the Score contest

SCRUM implementation and project plan

Project Plan

14/11/2015 to 28/11/2015: First Sprint + Alpha Version
28/11/2015 to 12/12/2015: Second Sprint + Beta Version
12/12/2015 to 26/12/2015: Third Sprint
26/12/2016 to 7/01/2016: Refinements, documentation upgrade
7/01/2016: Final Version

Scrum roles

- Product owner: Claudio Cardinale
- Scrum master: Giorgio Pea
- Development Team: All members of the project team

Development Team organization

Backend team:
Claudio Cardinale, controller/model developer, architecture manager
Emil Siladi, model developer
Mihovil Vinkovic, model developer
Denis Kuryshov, controller developer

Frontend team:
Giorgio Pea, view and logic developer
Jean Barre, view developer

Contact with the customer

The customer has been contacted only two times: the first time to discuss a target change for the application (the target was changed toward small organizations and businesses) and the second time to have an opinion about the final product. The communication has been carried out via email. This low frequency of communication has been caused by the low time available for the team and by the fact that the description of the project provided by the Score was clear and understandable

Daily meetings, sprint review meetings

The team has decided that during each sprint the daily scrum meeting will be held everyday at 18:30 CET/CEST (according to european DST)
The team has decided that at the end of each sprint a sprint review meeting will be held in the afternoon in a convenient time for each team member.

Activities performed before the starting of the first sprint

Before the starting of the first sprint, the project team has completed the following tasks:

- Requirements gathering and discussion
- Choice of communication tools, meetings frequency, sprints duration
- Choice of development roles
- Choice of SCRUM roles
- Choice of programming languages and technologies
- Choice of a suitable software architecture

Sprints

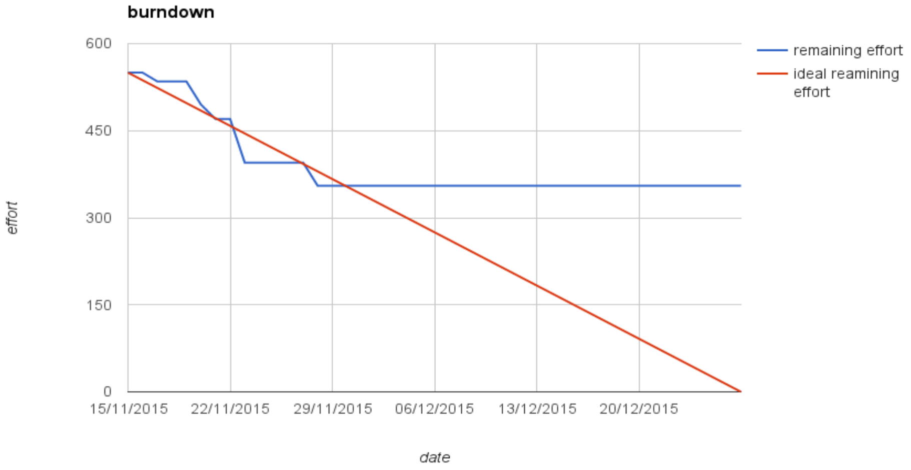
The project team has decided to make 3 sprints of a 2 weeks length, this length has been chosen because it guarantees a good balance between the necessity of being flexible and adapts to changes and the necessity of a sufficient quantity of time for development with quality and robustness the product.

Sprint 1 (from 14/11 to 28/11)

Requirements to be completed	**Was it completed?**
Organization Login and Registration (Reqs 1.1, 1.2,1.3)	YES
Organization’s members registration, management and login (Reqs 2.1, 2.3, 4.1, 4.2, 4.3)	YES
Organization’s teams creation and management (Reqs 3.1, 3.2, 3.3, 3.4, 3.5)	YES
An organization’s member can import his/her schedules from an external calendaring service using the CalDav protocol (Reqs 9.1, 9.2, 9.3, 9.4, 9.5, 9.6)	NO
A planner can plan a meeting for its associated teams (Reqs 10.1, 10.2, 10.3)	NO

Period	**Hours invested**

First Sprint - first week	155
First Sprint - second week	167
	TOT: 322



Comments to this sprint:

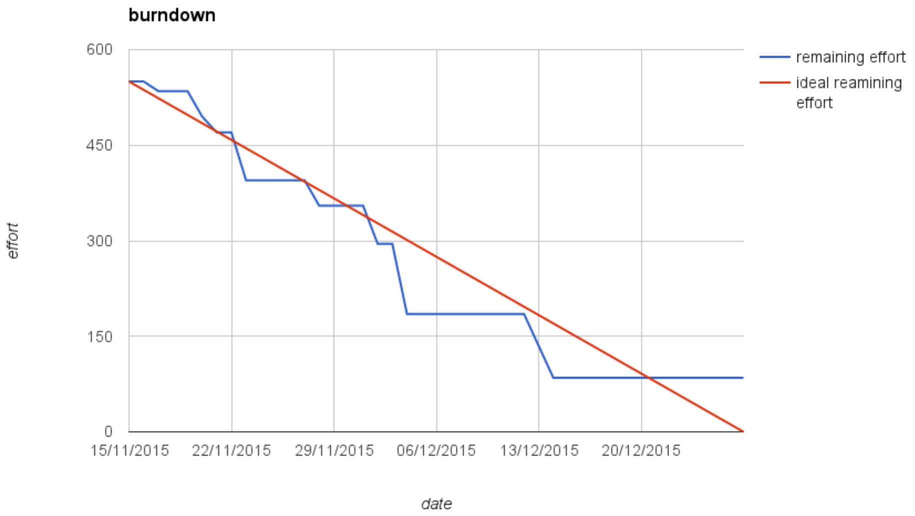
Some requirements have not been completed in time and have been delayed to the second sprint. After this first sprint the development team has realised that is necessary to increase the quantity of work and to be more synergetic.

Sprint 2 (from 28/11 to 12/12)

The following requirements have been added to the project: 1.8, 1.9, 4.5, 4.6
The following requirements have been dropped: Reqs 1.4, 1.6, 1.7, 2.2, 2.4, 3.6, 4.4, 5.1, 7.1, 7.2, 8.2, 9.2, 9.3, 9.4, 9.6, 10.4

The dropping of the above mentioned requirements has been decided for reasons of time and because of the common will of focusing on the meeting’s time optimization as the main feature of the product.
At the end of this sprint all the requirements of the software have been completed.

Period	**Hours invested**
Second Sprint - first week	196,5
Second Sprint - second week	224,5
	TOT: 421



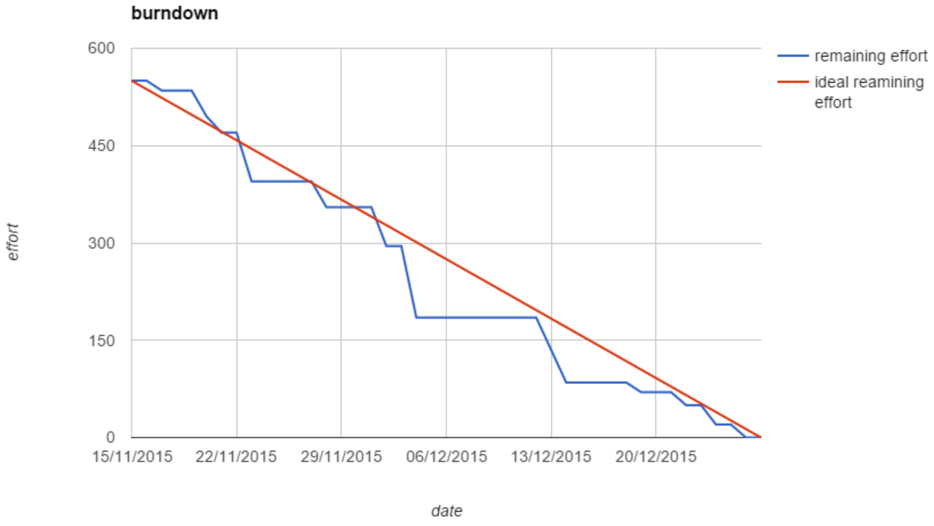
Comments to this sprint

Some lack of communications emerged during this sprint, nevertheless the team manages to complete all the requirements and also update some documentation.

Sprint 3 (from 12/12 to 26/12)

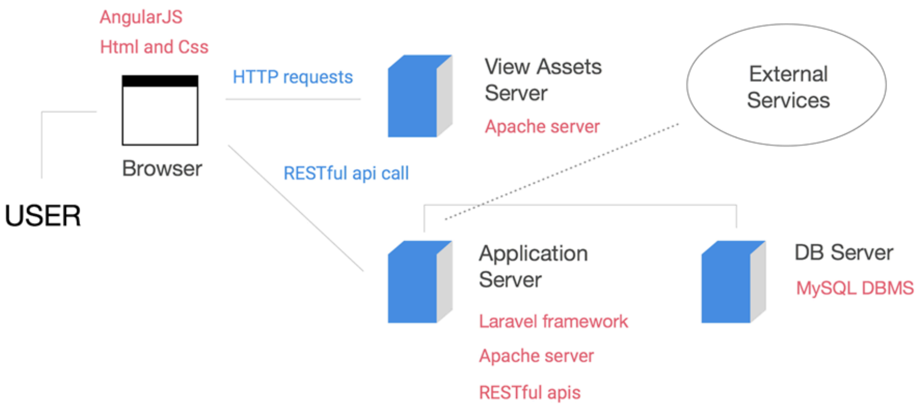
This sprint has been dedicated to testing, polishing and verification of the product. During this sprint the frontend team has completed the design of the web interface of Plunner in a way that it adapts to different screen resolutions (responsive design).

Period	**Hours invested**
Third Sprint - first week	81,5
Third Sprint - second week	91
	TOT: 172,5



Design and implementation

Architecture



As visible from the picture above, Plunner adopts a 3 tier* client server architecture, the main reasons behind this choice are the following ones:

- Scalability: the nodes composing the application server and db server can be increased in number without the need of redesigning the whole system
- Reliability: since the chosen architecture is distributed, failures can be easily isolated and database replications can be implemented without impacting too much on the design of the whole system
- Adaptability: the chosen architecture can be easily enriched with a load balancer that distributes and activates nodes in relation to the number of incoming requests.
- Security: since the chosen architecture strictly divides business logic and data and since the application server and the db server can be easily isolated from the web using firewalls, malicious attacks can be prevented and high level of security granted

**The web assets server is not considered in the account of tiers since its only purpose is to deliver via the web the code that after being executed by a browser implements the frontend part of Plunner*

Tiers

- Application Server: a tier that manages and implements the business logic of the entire application, it process and answers to requests from client's browsers
- DB Server: a tier that manages the persistent data of the application
- Browser: a tier that represents the layer of interaction between the client and the application

MVC

Both for the frontend and backend the team has decided to adopt a design pattern known as MVC (Model View Controller). This choice has been made since the pattern is one of the most used and tested in the industry and it lets the team split the work better and in a more systematic way.

Client Server Communication

The communication between the client and server is realised via a remote set APIs that the server makes available to the client. This communication method has been chosen for reasons of flexibility, security and because it is easier to implement.

External services

Since Plunner is a calendar based application, an integration with external calendaring services has been designed; the purpose of such integration is to give to the client the chance to easily import schedules and facilitate the meeting planning. The team has thought about using a standard and well diffused/supported protocol for integrating external calendaring services and so during the implementation phase this has been taken in consideration.

User interface

The team has decided to adapt a sleek, modern, minimalistic and business oriented web interface that must be responsive.

Technologies

Browser tier

HTML, CSS, Javascript, AngularJS framework

DB server

MySQL, AWS (as future improvement for High traffic)

Application Server

PHP, Laravel framework (LTS version), Apache Server, AWS (as future improvement for High traffic)

Client Communication

Remote set of apis implemented via the Laravel Framework following the RESTful standard

External Services

External calendaring services integration is made using the CalDAV.

The application respects user’s privacy and only reads the timeslots (meaning the starting and ending times of every timeslot) in the calendars.

These technologies were chosen by the team for the following reasons:

- They are easy to learn and use
- They are powerful and well diffused (standard open protocol)
- Some members of the team had previous experience with them

Optimization

Since the members of the project do not have only programming skills, we decided to use any available resources to perform a competitive product. Among them, knowledge in Operations Research was helpful to optimise Plunner.
By analyzing the computation of a starting meeting time according to each participant's schedules, we defined our problem through an linear program. We used GLPK, an open source program to resolve the optimization problem, maximising the number of people to attend the meeting. Below the mathematical model for implementation is detailed.
We maximise the number of people attending a meeting. We define two binary vars:
xij = {1 : person i attends meeting j; 0 : other cases}
yij = {1 : meeting i is planned to start at timeslot j; 0 : other cases}
In fact since we had to transform timeslots in a discrete form we decided to consider a timeslot like an entity with a fixed time, for example we can have the following time zones (every 15 minutes):
1 -> 01/12/15 00:00; 2 -> 01/12/15 00:1; 3 -> 01/12/15 00:30 ...
We have also the following parameters:

- UserAvailability: shows the timeslots when the user is available
- MeetingsAvailability: shows the timeslots when the meetings can be planned
- UsersMeetings: shows which users should participate to the meeting

We also have a list:

- MeetingsDuration: says the duration of a meeting

So we optimize: $$max \sum_{i \in Users;\, j \in Meetings} (x[i,\, j] \cdot MeetingsDuration[j]\,)$$

Here we have the complete model (with all constraints):
https://github.com/dsd-meetme/backend/blob/master/app/Console/Commands/Optimise/model.stub

With this optimization we are able to plan all meetings automatically, but to do that we have to add some constraints (these are not the linear programming constraints) to work in a real environment.

- A planner can plan a meeting for the weeks after, but a meeting must be planned inside one week
- The optimization task is done on Sunday at midnight for the next week then automatic email notifications are sent
- The week starts on Monday
- A planner can plan a meeting for the next week until Sunday midnight, since the optimization is done on Sunday at midnight for the next week
- After the optimization is not possible to add new meetings to that week, but it is possible to remove and update (information not date)
- A user can update the busy timeslots for the week after until Sunday midnight, like planning. He can always compile busy timeslots for the weeks after the next one
- Timeslot is multiple of 15 minutes
- We always use UTC time

We have to proceed in this way since we decided to perform the optimization task one time each week and it optimises the week after.

Counterexamples:

- if we select a longer period, meetings with availability of the next week are often planned for it (probably in the next week they are able to find more employees), but at the same time on the week after we have some meetings that are already planned and we cannot optimise them again, so we cannot optimise the week properly because we have old meetings planned that are like busy timeslots. If we don't plan them we are able to optimise the entire week with them (shifting them)
- if we select a lower period we don't have enough timeslots to perform a good optimization in fact for example if we choose a period of 3 days we have few timeslots available to shift meetings and we don't have the best optimization

Improvements:

- Customization: the company can choose the frequency of optimization task
- At the moment we can have meetings with only one employee (useless)
- Different kind of users (required and invited) given priority to one type
- Concentrate meetings to avoid empty timeslots
- If we have optimization errors -> rollback for that company
- Concentrate meetings to avoid empty timeslots

Future improvements

Other external calendar services support

Users would appreciate a possibility to link their accounts with specific calendaring application like Google Calendars or Microsoft Exchange through their specific APIs.

LDAP or Lightweight Directory Access Protocol

It is widely used by companies to manage their employee's data. With this feature Plunner could be used not only by little structure organisations but also by the big companies.

HTTPS

HTTPS greatly increases the security of the application a modern application should have it.

Future Plunner architecture with mobile application

The Plunner architecture in future can include such external additions as Mobile application. Since during our development we have used RESTful api, it is very easy for extending.

Improve testing

Increasing test coverage, making automated whole system tests and increasing the amount of stress testing done.

Unimplemented requirements

Due to timing restraints some requirements had to be cut.

Meeting prioritization

We want meetings to have a priority associated with them, which would make choosing to which meeting to assign to which user, if there are meeting conflicts, easier.

Verification and validation

The verification and validation of Plunner has been conducted by the development team using automatic tests and manual test with the support of different automated tools, such as:

- Travis CI for continuous integration with different environments
- Scrutinizer for code quality checking and test coverage (for Backend)
- CodeClimate for code quality and test coverage (for Frontend)

Every backend and frontend team member has worked in its own branch. After successful testing, they pushed their results into master branch.

Frontend

Unit tests

Because of the low time available for the process and the relative complexity of it, unit testing for the frontend covers only some modules of the application, the focus has been pointed on manual tests. The tools used for frontend unit tests are Karma and Jasmine.

The results of frontend tests can be seen in the following links:

- For Code Climate: https://codeclimate.com/github/dsd-meetme/frontend
- For Travis CI: https://travis-ci.org/dsd-meetme/frontend

Manual tests

The manual tests performed by the frontend team include:

- Responsiveness tests: testing if the web interface adapts to different screen sizes
- Requirements testing: testing if all the functionalities exposed by the frontend part of the application cover all the requirements
- Integration testing: testing if the frontend part integrates perfectly with the backend.
- Browser testing: testing if the web interface works properly with different browsers

Backend

Unit tests

The tool used for backend unit tests is PHPUnit customized by laravel. Due to development in backend team, all participants of it performed backend tests.

The results of backend tests can be seen in the links:

- For Scrutinizer: https://scrutinizer-ci.com/g/dsd-meetme/backend/?branch=master
- For Travis CI: https://travis-ci.org/dsd-meetme/backend

Acceptance test plan

A simple acceptance test plan has been made so that the application is accepted if:

- It guarantees all the requirements
- It has a responsive web interface
- Its frontend part supports different browsers (including mobile browsers)

Lessons learned

Things that should have been improved

- Communication among team members
- Testing: more modules of the frontend part of the project should have been unit tested and end to end tests using a tool like protractor should have been implemented
- Optimization constraints on backend side

Things well done

- The team has used the Laravel framework version 5.1 LTS (3 years of support), so that the software remains stable and well supported
- All errors are logged
- The project's files organisation ideal for a big project
- The project can be easily configured via config files
- People testing the entire environment are different from people that developed these functionalities to give an independent way
- A simple presentation site has been made to present the product and link to installation instructions
- All time data is converted in UTC for reasons of simplicity and readability
- To prevent malicious attacks, every 30 days a user of Plunner has to sign in again even if he or she has turned on the remember me functionality
- The frontend and the backend part of the application have been developed using two repositories, so that the designed architecture of Plunner is perfectly reflected. This organization of things accelerates and facilitates the development process
- Use of external services for high resources task, like sending emails
- As authentication technology JWT has been used: it protects against attacks like CSRF or XSS and allows to reduce the amount of resources need on the backend server.
- The backend part of the application is packeged via https://packagist.org/